Nome: Bernardo e Bryan

1) Várias foram as “falhas históricas de desenvolvimento de software” que geraram prejuízos. Faça uma pesquisa e cite exemplos de falhas descritos detalhadamente (mínimo 3 exemplos).

1- Caso do *Therac-25* (1985-1987)

O *Therac-25* era um equipamento de radioterapia utilizado para tratar pacientes com câncer, diversos incidentes ocorreram devido a falhas no *software* que resultaram na superexposição de radiação, causando sérios danos e até mesmo mortes. O software do Therac-25 tinha um *bug* que permitia doses de radiação muito maiores do que as seguras fossem administradas. Além disso, o sistema não foi testado adequadamente para garantir que erros de software não comprometeriam a segurança do paciente e a resposta dos desenvolvedores e da empresa à ocorrência dos acidentes foi lenta e insuficiente faltando um processo robusto de revisão e uma abordagem proativa. Os erros impactaram em pelo menos seis mortes e muitos pacientes com graves danos, o caso levou a uma revisão critica das práticas de engenharia de software e a um aumento na conscientização sobre a importância de testes e verificações rigorosas para a segurança.

2- Bug do Milênio (Y2K) (1999-2000)

Ocorrida no final do milênio – origem do nome –, esta falha estava relacionada à forma com que os sistemas armazenavam as datas nos computadores. Diversos sistemas utilizavam dois dígitos para o ano. Tal maneira simplória poderia levar a mudança para o ano 2000 (00) a ser interpretada como uma transição para o ano 1900. Apesar do pânico à época, os efeitos não foram tão catastróficos como esperados, embora diversos sistemas de áreas diversas tenham sido de alguma maneira afetados. O motivo para o efeito ter sido menos destrutivo não foi o acaso. Os custos para frear o evento variaram de bilhões a trilhões de dólares globalmente. O legado do Y2K permanece até hoje, lembrando desenvolvedores a não subestimarem informações consideradas "simples", bem como a incorporar práticas mais rigorosas de verificação e teste de software.

3- Incidente do “*Mars Climate Orbiter*” (1999)

O *Mars Climate Orbiter* era uma sonda da *NASA* lançada para estudar o clima de Marte. Em setembro de 1999, a sonda se desintegrou na atmosfera de Marte devido a um erro de cálculo de trajetória. O principal problema foi um erro de unidade de medida. Fora usado no desenvolvimento libra-força (uma unidade imperial) em vez de *Newtons* (a unidade métrica) para calcular as forças aplicadas à nave. Houve também problemas de comunicação entre diferentes equipes de desenvolvimento e entre contratados e a *NASA* – tal falta de integração e comunicação contribuiu para a falha de implementação. Um total de aproximadamente 327 milhões de dólares foram desperdiçados devido ao erro. O incidente destacou a importância de padrões rigorosos de qualidade e de verificar a consistência nas unidades de medida, além de melhorar a comunicação entre os envolvidos em projetos complexos.

2) Qual foi o motivador para o surgimento da Engenharia de Software?

Nas décadas de 1960 e 1970, diversos fatores acumulados, como crescente complexidade, problemas de qualidade, atrasos e custos exorbitantes e difícil manutenção, levaram à chamada Crise de Software. Tal evento escancarou a necessidade da adoção de algum tipo de metodologia de produção, bem como a de uma padronização de desenvolvimento e testes.

Em meio a esse contexto, foi cunhado então, na Conferência de Engenharia de Software da OTAN em 1968, o termo Engenharia de Software. O objetivo do campo era simples: dar um tratamento de engenharia (mais padronizado e controlado) ao desenvolvimento de software.

3) Em duplas, monte uma linha do tempo sobre a evolução da engenharia de software desde os anos 1950 até hoje.

Linha do tempo da Engenharia de Software.

Primeira Era (1950-1960)
- O conceito de software começou a ganhar popularidade com a vinda dos primeiros computadores eletrônicos.

Segunda Era: A crise do Software (1960 – 1970)
- Com o desenvolvimento de softwares mais complexos e críticos, surgiram problemas significativos como atrasos nos prazos, estouros de orçamento, software de baixa qualidade e falhas catastróficas. A crise levou à necessidade de metodologias mais estruturadas para o desenvolvimento de software, originando o campo da Engenharia de Software.

Terceira Era: Metodologias Estruturadas (1970 – 1980)
- Surgimento de metodologias como o Waterfall (Modelo Cascata), organizava o desenvolvimento de software em etapas sequenciais: requisitos, design, implementação, testes e manutenção. Com a Engenharia de Requisitos, deu início a formalização do processo de levantamento de requisitos e design de sistemas, dando reconhecimento a importância de capturar corretamente as necessidades dos usuários antes de iniciar o desenvolvimento.

Quarta Era: Reuso (1980 - 1990)
- Começou a ser usada a reutilização de componentes de software como uma maneira de aumentar a produtividade e melhorar a qualidade, levando ao desenvolvimento de bibliotecas e frameworks.

Quinta Era: Metodologias Ágeis e Desenvolvimento Iterativo (1990 - 2000)
- Surgiram metodologias ágeis (ex.: Scrum, XP) que enfatizam entregas incrementais e iterativas, colaboração contínua com o cliente, e flexibilidade para mudanças nos requisitos.

Sexta Era (2000 - Presente)
- Práticas de Integração e Entrega contínuas permitem que equipes integrem e testem mudanças de códigos de forma contínua. Isso resulta em ciclos de desenvolvimento mais rápidos e melhora a qualidade do software.